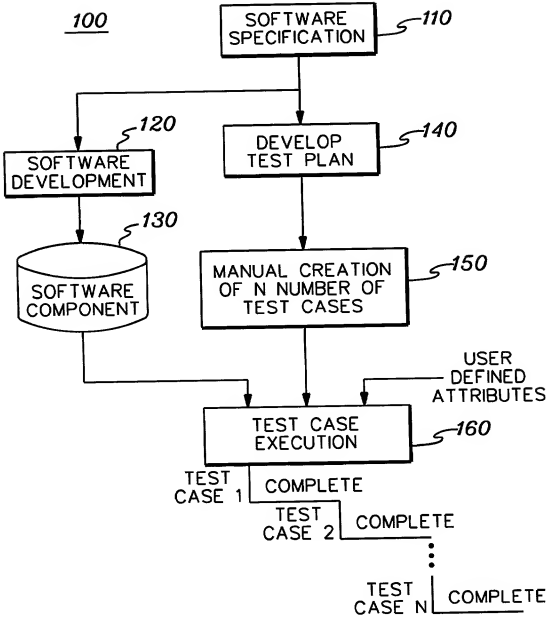

fig. 1
(PRIOR ART)

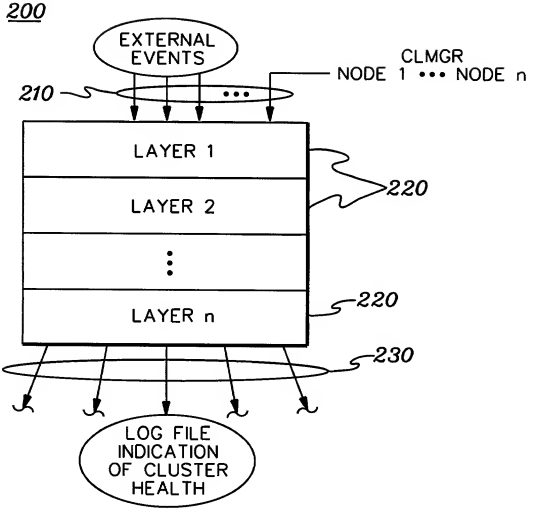

fig. 2

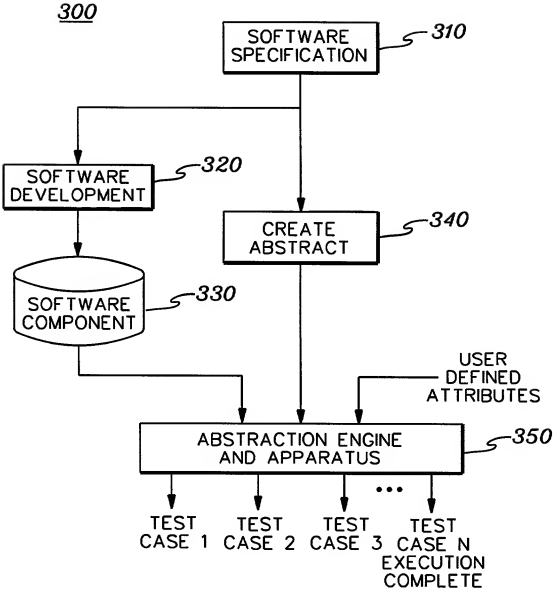

fig. 3

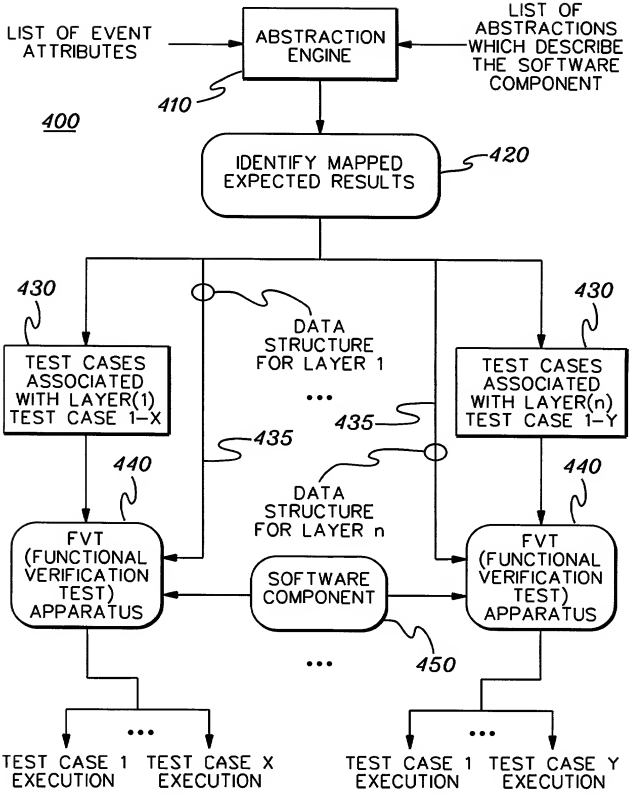

fig. 4

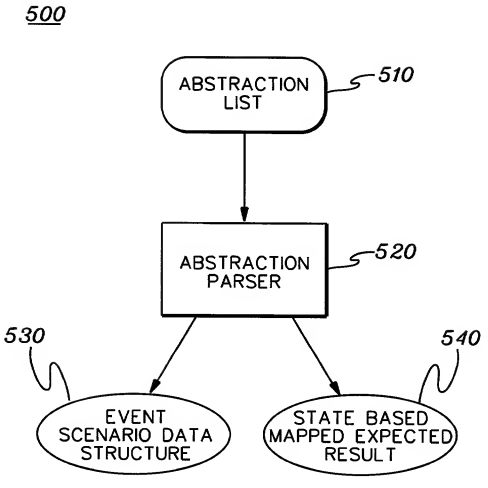

fig. 5

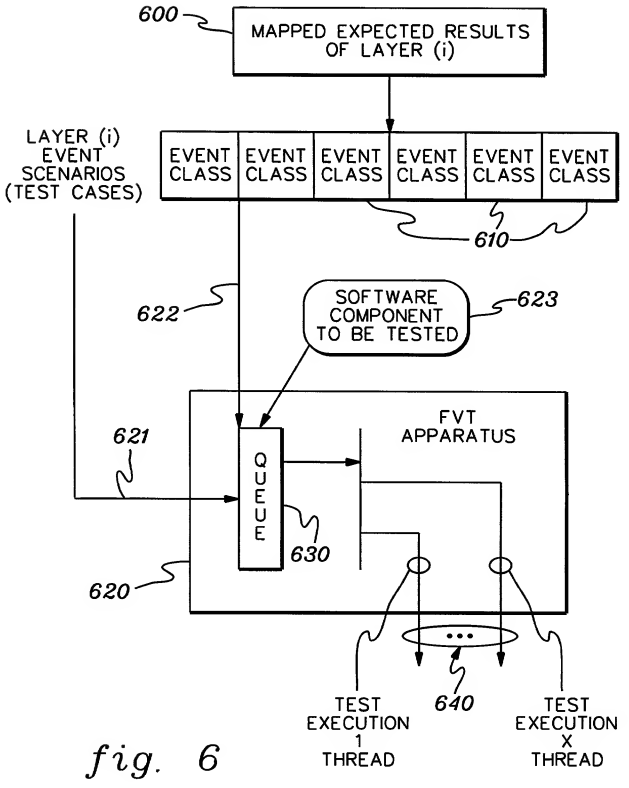

fig. 6

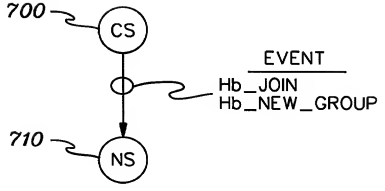

fig. 7

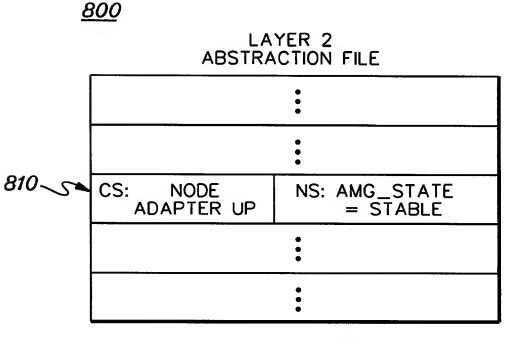

fig. 8

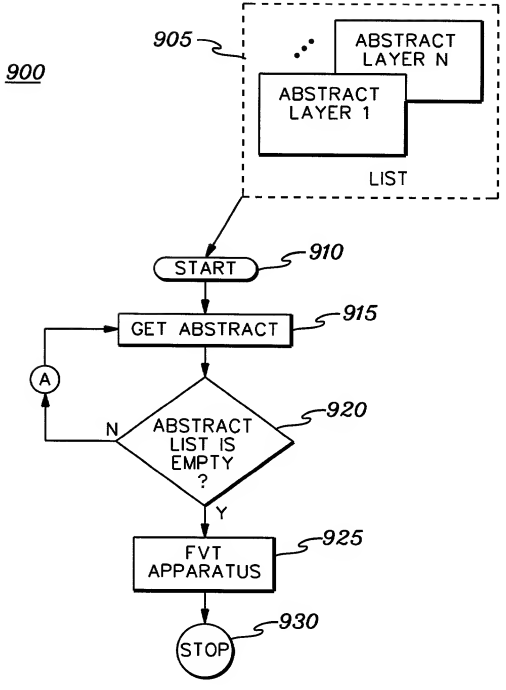

fig. 9A

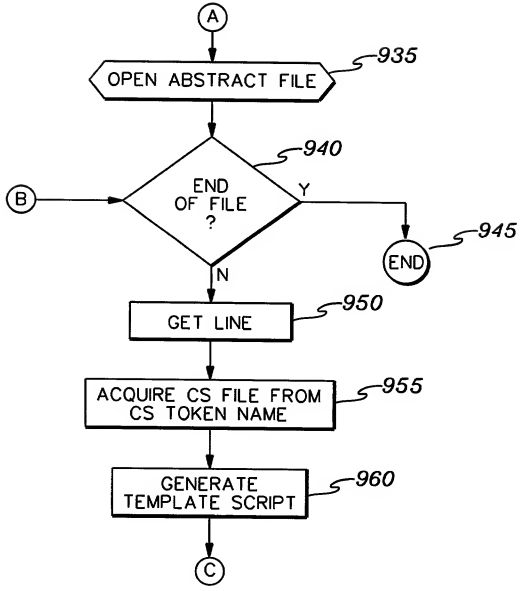

fig. 9B

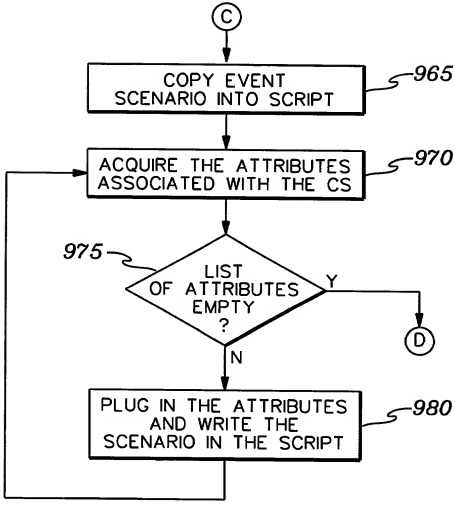

fig. 9C

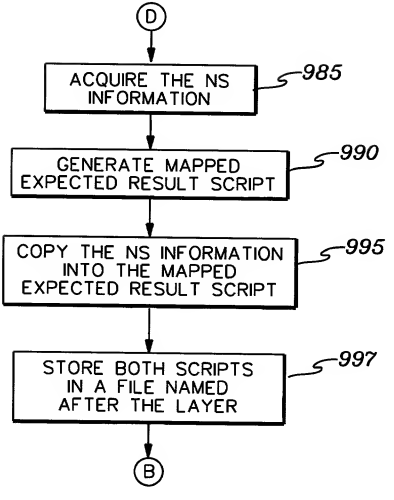

fig. 9D

PSUEDO_CODE. TXT

PSEUDOCODE FOR THE ABSTRACTION ENGINE AND POV (POINT OF VERIFACATION)
EACH LAYER HAS ITS OWN ABSTRACT FILE.

OPEN FILE. FOR EACH LINE DO THE FOLLOWING

READ CS: TOKEN (CURRENT STATE)
READ NAME AND ACQUIRE THE DATABASE (FILE CONTAINING THE DETAILED EVENT SCENARIO, THAT WILL TAKE YOU TO THE "NEXT STATE")
(THIS INFORMATION IS THE SYNTAX AS DESCRIBED IN THE SYSTEM DESIGN)

AN EXAMPLE IS Hb_JOIN <ADAPTER> WHERE ADAPTER IS A VARIABLE WHICH WILL BE FILLED IN LATER
CREATE A TEMPLATE KSHELL SCRIPT,
COPY THE EVENT SCENARIO FROM THE DATABASE,
PLUG IN THE ATTRIBUTES AND CREATE AN EVENT SCRIPT
REPEAT UNTIL YOU HAVE REACHED THE END OF THE ATTRIBUTE LIST
EXAMPLE Hb_JOIN en0 WHERE en0 IS AN ADAPTER TAKEN FROM THE ATTRIBUTES LIST WHERE EACH SCENARIO CONSTITUTES A TEST CASE

READ NS: TOKEN

CREATE PERL SCRIPT TEMPLATE, NAME IT AFTER THE NS TOKEN NAME
EXAMPLE <AMG_STATE STABLE>
MODIFY PERL SCRIPT WITH THE EVENT SCRIPT(S)
READ NAME AND ACQUIRE THE DATABASE (FILE)
WHICH CONTAINS THE DETAILED SCENARIO OF WHAT THE "NEXT STATE" IS
(THIS INFORMATION IS WHAT SHOULD BE CONTAINED IN THE SYSTEM DESIGN DOCUMENT. ALSO THIS INFORMATION WAS USED AS PART OF THE REVIEW BY THE TESTER, AND THE COMPONENT DEVELOPER)
CREATE THE EVENT CLASS (MAPPED EXPECTED RESULTS) USING THE DETAILED INFORMATION ABOVE, AND THE ATTRIBUTES.

STORE THE TESTCASE PERL SCRIPT AND EVENT CLASS, IN DATABASE/FILE NAMED AFTER THE LAYER.

fig. 10

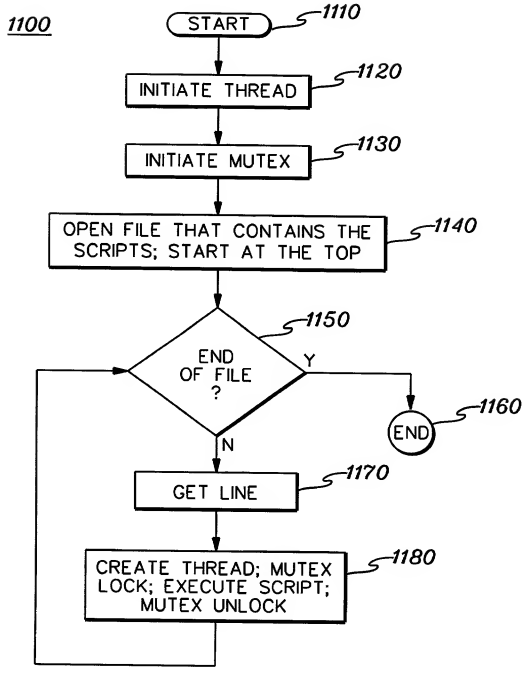

fig. 11

```
/* PseudoCode for fvt apparatus */

#include <pthread.h>
#include <stdio.h>

/* This is the initial thread routine */
void* compute_thread(void*);

/* This is the lock for thread synchronization */
pthread_mutex_t my_sync;

/* This is the condition variable for task order control */
pthread_cond_t rx;

/* This is the Boolean */
int thread_done = FALSE;

main()
{

/* This is data describing the thread created */
pthread_t tid;
pthread_attr_t attr;

/* Start of executable */

/* Initialize the thread attributes */
   pthread_attr_init(&attr);

/* initialize the mutex (default attributes) */
   pthread_mutex_init(&my_sync, NULL);

/* initialize the condition variable (default attributes) */
   pthread_cond_init(&rx, NULL);
```

fig. 12A

```
/* Create another thread. The Thread ID is returned in &tid */
/* The last parameter is passed to the thread function */
   while (the file containing the testcases for a particular
layer is not empty do the following)
{

pthread_create (&tid, &attr, compute_thread, invoke_perlscript);

 /* wait until the thread does its work */
    ptrhead_mutex_lock(&my_sync);
    while (!thread_done) pthread_cond_wait (&rx,&my_sync);

 /* When we get here, the thread has been executed */
    printf(thread);
    printf("\n");
    pthread_mutex_unlock(&my_sync);
    exit(0);
    } /* end of do while
    } /* end of main routine

 /* The thread to be run by create_thread */
 void* compute_thread(void* invoke_perlscript)
 {
 /* Lock the mutex when its our turn */
    pthread_mutex_lock(&my_sync);

 invoke_perlscript
 /* set the predicate and signal the other thread */
 thread_done = TRUE;
 pthread_cond_signal(&my_sync);
 pthread_mutex_unlock(&my_sync);

 return;
 }
```


fig. 12B

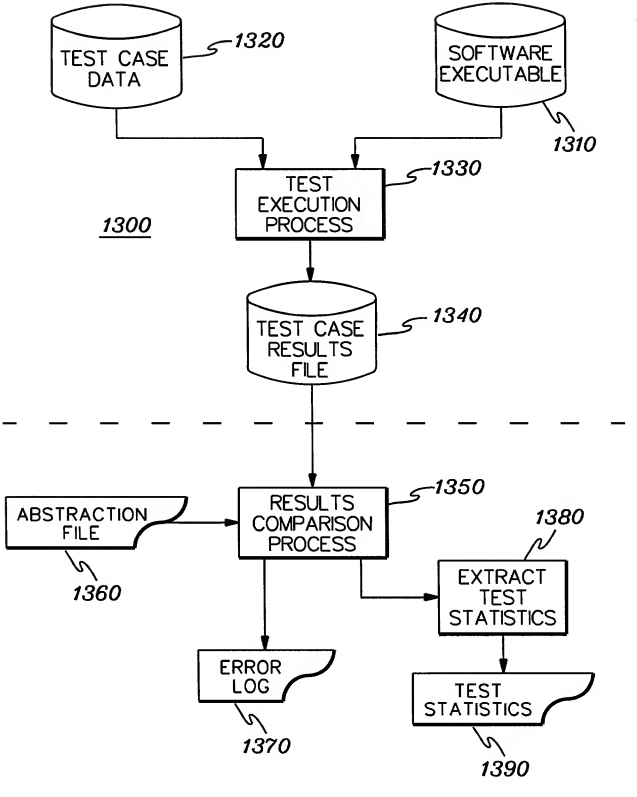

fig. 13

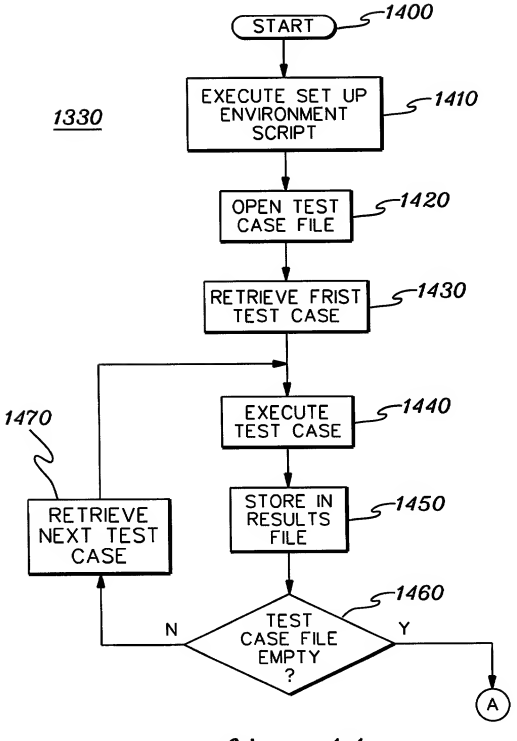

fig. 14

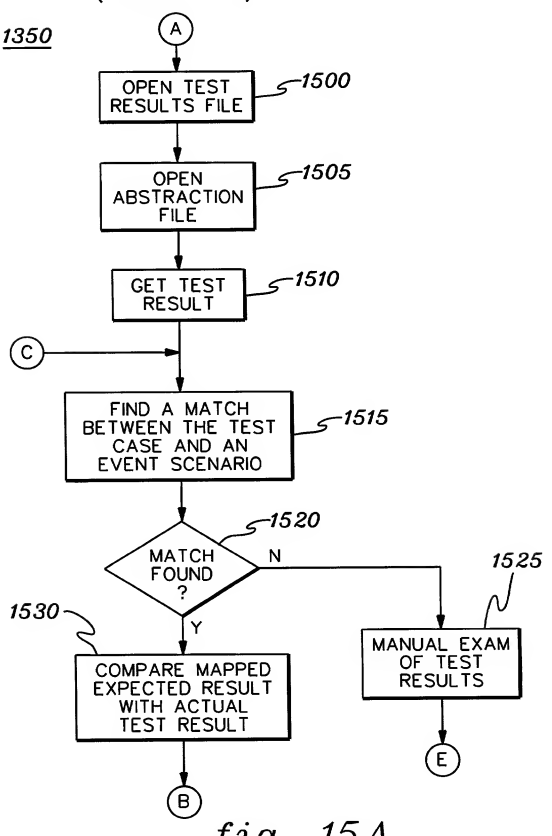

fig. 15A

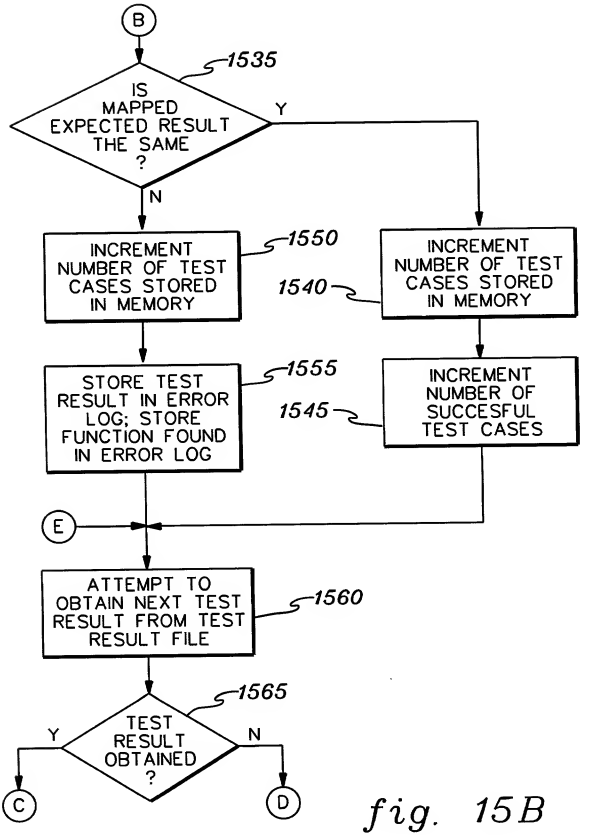

fig. 15B

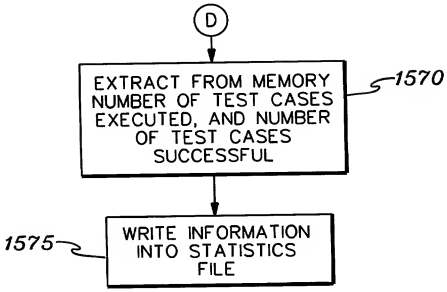

fig. 15C

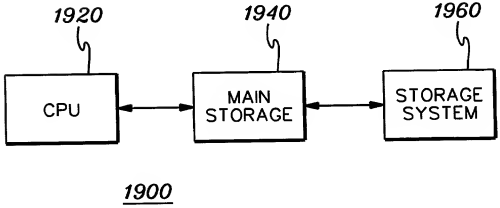

fig. 19

```
open(TR,"/test_results_file")

While(<TR>)
{

    get line of test result file
    store into temp file
    Get 3(superscript:rd ) token of test result file
    open(AB,"Abstract_file")
      While(<AB>) ( 3(superscript:rd ) token of test result
file is not equal to the NS name ) & (not end of file)
                  { get  next line of AB}
                 else (if 3(superscript:rd ) token of test
result file is not equal to the NS name)
                  {
                  open name of the file which will contain the
mapped expected results.
                 get 2(superscript: nd) token of test result
file
                 2(superscript:nd ) token of test result file is
the number of the entry into the table of the mapped expected
result

                 while(<TR>) (1(susperscript: st) token is not
equal to Result:)
                       {get next line of TR
                        store into temp file
                       }
                       else
                       {

                        get next line(s) of TR store into
temp file (this could be several lines) until deliminter**********
is reached
                        if contents of temp file equals"
Expected result" of table then
                                        {
                                          increment
Numver_of_test_cases_executed
                                          increment
Number_of_test_cases_successful
                                        }
                                else
                                        {
                                          increment
Number_of_test_cases_executed
                                          concatenate
```

fig. 16A

```
temp file into Error log
                                        }
                    }
                }
                else no match was found, list the test case
                close (AB)
                reset te mp file
    }

     close(TR)
      print<Date Number_of_test_executed
Number_of_test_cases_successful
Number_of_testcases_expected_successful>
```

fig. 16B

sr_create_directory+Treehandle+DirectoryName+StorageFlag+Force

TreeHandle	DirectoryName	StorageFlag	Force	Expected Result
VALID	ABSOLUTE	1	0	SR_SUCCESS
VALID	ABSOLUTE	1	1	SR_SUCCESS
VALID	ABSOLUTE	0	0	SR_SUCCESS
VALID	ABSOLUTE	0	1	SR_SUCCESS
VALID	ABSOLUTE	X	0	SR_SUCCESS
VALID	ABSOLUTE	1	INVALID	SR_INVALID_FORCE
VALID	RELATIVE	0	INVALID	SR_INVALID_FORCE
VALID	RELATIVE	1	0	SR_SUCCESS
VALID	RELATIVE	0	0	SR_SUCCESS

fig. 17

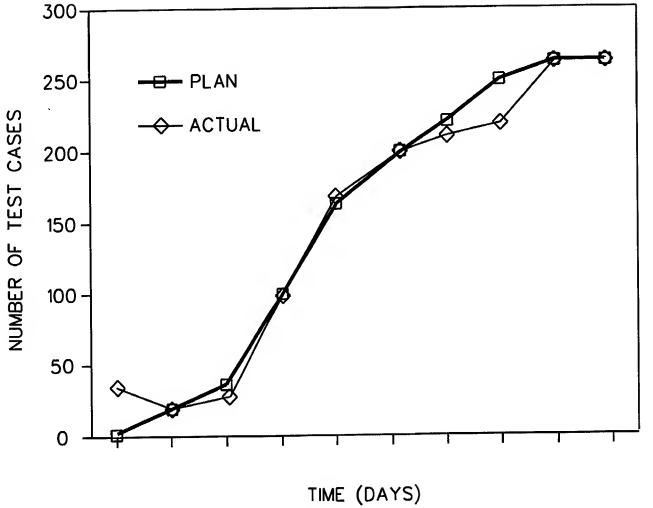
300
250
200
150
100
50
0
PLAN
ACTUAL
NUMBER OF TEST CASES
TIME (DAYS)

fig. 18